МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГУ»)

Факультет Компьютерных Наук

Кафедра программирования и информационных технологий

Клиент – серверное приложение «Wish Box»

Курсовой проект

090304 Программная инженерия

Выполнили:

Агафонова Марина Владимировна, студентка 3 курса, 4 группы
Аленичев Александр Викторович, студент 3 курса, 4 группы
Семечев Данила Алексеевич, студент 3 курса, 4 группы

Проверил:
Тарасов Вячеслав Сергеевич, ассистент кафедры ПиИТ

Воронеж 2020

Отчёт по 2 аттестации.

В течение второй аттестации было сделано следующее:

Агафонова Марина:

- Back-end:
  - Настройка проекта и подключение swagger
  - Классы моделей
  - Поиск пользователя с возможностью применения фильтра
  - Добавление нового желания (добавление UserId и изображения)
  - Добавление аватара и его редактирование
  - Редактирование желания (замена изображения)
  - Список желаний пользователя
  - Главная страница (вывод записей о добавлении нового желания пользователей из списка подписок текущего пользователя)
  - Пометка “Подарено” у желания
- Front-end:
  - Layout для всех страниц
  - Страница для неавторизованного пользователя
  - Страница авторизации
  - Страница регистрации
  - Страница добавления нового желания
  - Страница редактирования желания
  - Страница с желаниями пользователя
  - Страница поиска пользователя
  - Главная страница
  - Страница редактирования профиля
  - Страница пользователя
- Диаграммы:
  - Use case
  - Диаграмма последовательности
  - Диаграмма состояния
- Загрузка приложения в Azure

Аленичев Александр:

- Back-end:
  - Редактирование профиля
  - Добавление желания (основная часть)
  - Редактирование желания (основная часть)
  - Удаление желания
  - Изменение пароля
  - Добавление комментария
  - Удаление комментария
  - Создание комментария в ответ другому комментарию
  - Список тех подарков, которые взял на себя текущий пользователь
  - Подключение аналитики к проекту
- Front-end:
  - Страница изменения пароля
  - Открытие формы с комментариями
  - Список тех подарков, которые взял на себя текущий пользователь
- Аналитика:
  - Подключение аналитики «Яндекс Метрика»

    - Воронка «Регистрация»
    - Воронка «Создание желания»
    - Воронка «Подписаться»
    - Воронка «Добавить комментарий»
    - Воронка «Отметка “Подарить”»
    - Воронка «Отметить как “Подарено”»
- Диаграммы:
    - Диаграмма развёртывания
    - Диаграмма (объектов)

Семечев Данила:

- Back-end:
    - Настройка маршрутизации
    - Система авторизации и регистрации
    - Модель регистрации и авторизации
    - Рейтинг желаний пользователя
    - Класс модели WishRating + добавления в модели данных, необходимых для рейтинга
    - Страница профиля пользователей
    - Модель страницы профиля пользователей
    - Страница "мои подписки"
- Front-end:
    - Базовая вёрстка страницы профиля
    - Базовая вёрстка страницы «мои подписки»
- Диаграммы:
    - Диаграмма классов
    - Диаграмма взаимодействия
    - Диаграмма активности
- Аналитика:
    - Воронка «Авторизация»
    - Воронка: «Переход на страницу пользователя»
    - Воронка: «Оценка желаний»

Содержание

Введение

Все мы любим дарить и получать подарки. В этом есть что-то магическое. Но иногда бывает так, что полученный подарок вам не нравится, да и вы уже начинаете понимать, что подарили другому человеку не самую нужную или приятную для него вещь.

В таких случаях на помощь приходят всяческие сервисы подбора подарков, где другие люди уже составили какие-то подарочные наборы. Правда, обычно это бывает дорого и не всем по карману.

Также есть различные приложения по поиску подарков. Но хорошего приложения на данный момент не существует, а существующие не имеют нужной функциональности.

В данной курсовой работе рассматривается проблема создания веб-приложения, предоставляющего пользователям возможность публиковать список желаемых ими подарков, а также возможность дарить эти самые подарки другим людям.

1. Постановка задачи

Цель курсовой работы – реализовать клиент – серверное web приложение, позволяющий пользователям «делиться» своими пожеланиями в качестве подарков, а также позволяющий дарить желаемый предмет. Оно должно позволять пользователю:

- Создавать/редактировать/удалять своё “желание”.
- Отмечать подарок другого пользователя, который они хотят ему подарить, и удалять данную отметку.
- Подписываться через поиск по пользователям на другие профили и отписываться от получения новостей о добавлении новых желаний определённого пользователя.
- Оставлять комментарии к записи с желанием и удалять его.
- Изменять данные своего профиля, включая смену пароля.

2. Анализ предметной области

2.1 Глоссарий

- Желание (подарок) - запись, которую пользователь публикует на странице своего профиля в качестве отображения того, что он хотел бы получить в подарок.
- Подписчик - пользователь, чьи новые желания будут видны на главной странице текущего пользователя.

2.2 Анализ существующих решений

Рассмотрим существующие решения.

2.2.1 Gmoji

Само приложение — это витрина товаров и услуг, которые можно приобрести в подарок. Выбор большой: от кофе и цветов до мойки машины и стрижки. Всего разделов три: каталог, где можно приобрести подарки, профиль пользователя и список приобретенных и полученных подарков. Посылать подарки друзьям можно как из приложения, так и через мессенджеры, если включить в настройках клавиатуру Gmoji. Отправленные ссылки превращаются в стикеры с крупной картинкой, а в других мессенджерах выглядят просто как ссылки.

Плюсы:

- приятный дизайн и интуитивно понятный интерфейс приложения
- огромный выбор подарков
- не требуется наличие приложения для получателя подарка

Минусы:

- отсутствие версии для браузеров
- не самая удобная система “дарения” подарков через подключение еще одной клавиатуры на телефоне

2.2.2 WishBox

Приложение доступно в AppStore и Google Play. Разрабатывает его небольшая команда разработчиков. Приложение позволяет добавлять свои подарки, чтобы люди знали, что вам дарить, а также отмечать подарки других людей, чтобы вы не забыли

Плюсы:

- Возможность создавать свои списки желаний, а также событий, к ним причастных
- Возможность добавлять людей в “друзья”
- Новостная лента, чтобы всегда видеть чего твои друзья хотят
- Уведомления
- Возможность добавлять желания друга в свои

Минусы:

- Периодически происходят сбои сервера
- Интуитивно непонятный интерфейс
- Функционал на разных платформах различается
- Невозможность фильтра новостной ленты
- невозможность фильтра результата поиска (т.е. поиск только по имени пользователя)

2.3 UML диаграммы

2.3.1 Диаграмма вариантов использования

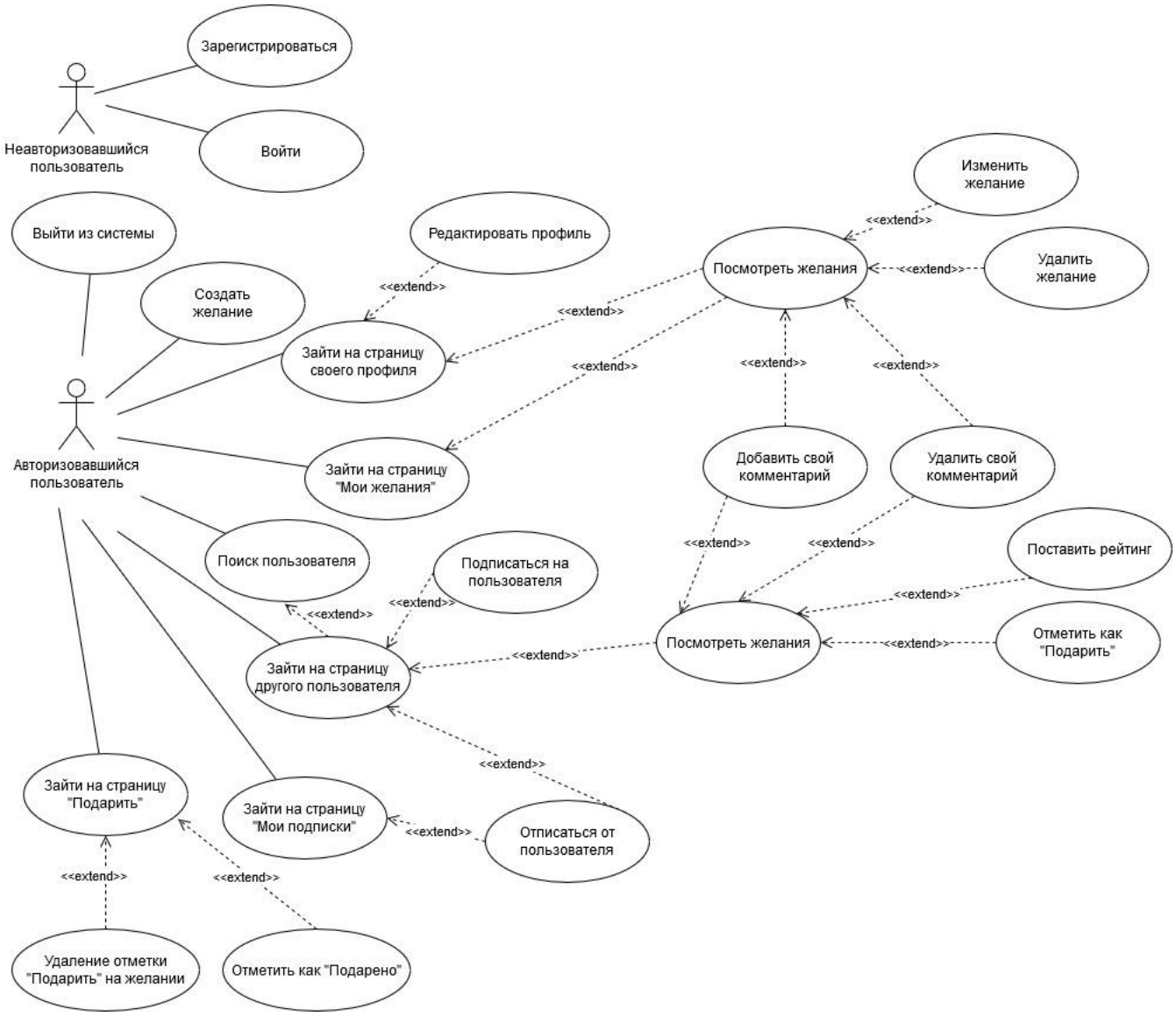

Рисунок 1. Диаграмма вариантов использования

Рассмотрим диаграмму вариантов использования для приложения на рисунке 1.

В системе могут быть два вида актора:

- Неавторизовавшийся пользователь
- Авторизовавшийся пользователь

Отношения ассоциации – действия, которые может осуществлять пользователь.

Для неавторизовавшего пользователя отношения ассоциации:

- Зарегистрироваться
- Войти

Для авторизовавшегося пользователя отношения ассоциации:

- Выйти из системы
- Создать желание
- Зайти на страницу своего профиля
- Зайти на страницу «Мои желания»
- Зайти на страницу другого пользователя
- Зайти на страницу «Мои подписки»
- Зайти на страницу «Подарить»
- Найти пользователя в системе по имени

Для авторизовавшегося пользователя отношения расширения:

- Редактировать свой профиль
- Изменить своё желание
- Удалить своё желание
- Добавить комментарий
- Удалить свой комментарий
- Поставить рейтинг желанию другого пользователя
- Поставить отметку «Подарить» на желание другого пользователя
- Удалить отметку «Подарить» у ранее отмеченного желания.
- Поставить отметку «Подарено» на желание, которое есть на странице «Подарить»
- Подписаться на другого пользователя
- Отписаться от пользователя, на которого был ранее подписан текущий пользователь.

2.3.2 Диаграмма последовательности

Диаграмма последовательности для создания желания, рисунок 2

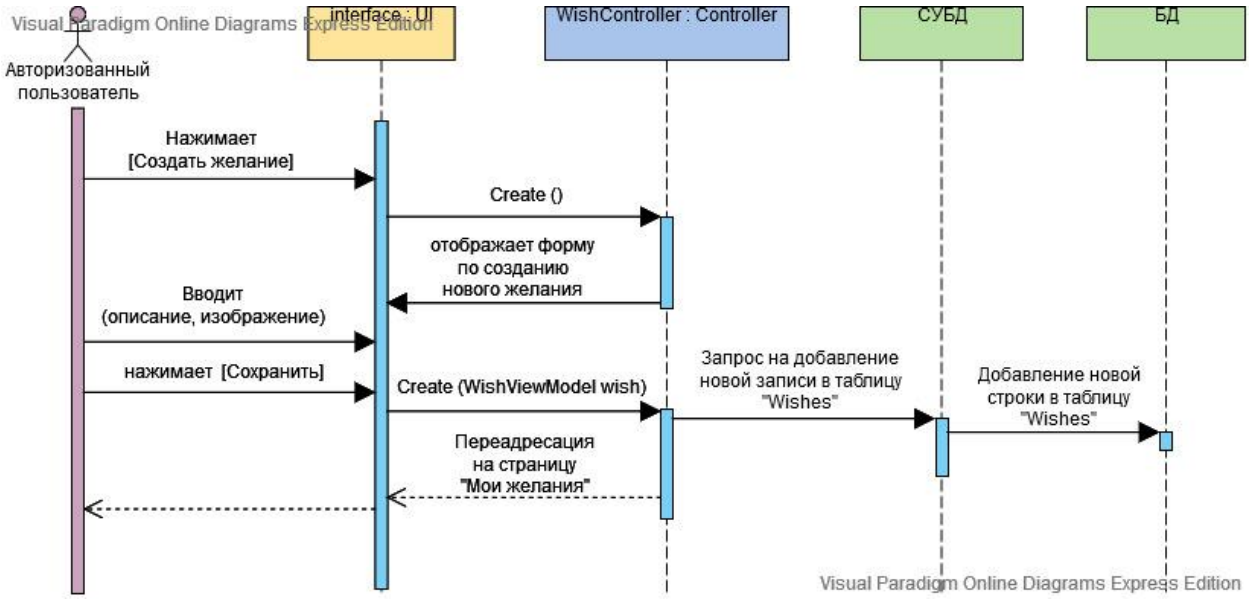

Рисунок 2. Диаграмма последовательности для создания желания

После взаимодействии пользователя, который обязательно должен быть авторизован, с интерфейсом приложения (нажатием соответствующей кнопки для создания желания) вызывается метод Create в контроллере, который возвращает форму по созданию нового желания. После введения описания желания и опциональной (необязательной) загрузки изображения для желания и нажатия кнопки «сохранить» вызывается метод Create (POST-запрос) с аргументом WishViewModel – моделью просмотра желания, которая содержит все данные, внесённые пользователем. Отравляется запрос в СУБД на добавление новой записи в таблицу «Wishes». В результате добавляется новая строка в таблицу «Wishes» в базе данных. Происходит переадресация пользователя на страницу «Мои желания».

Диаграмма последовательности для создания желания, рисунок 3

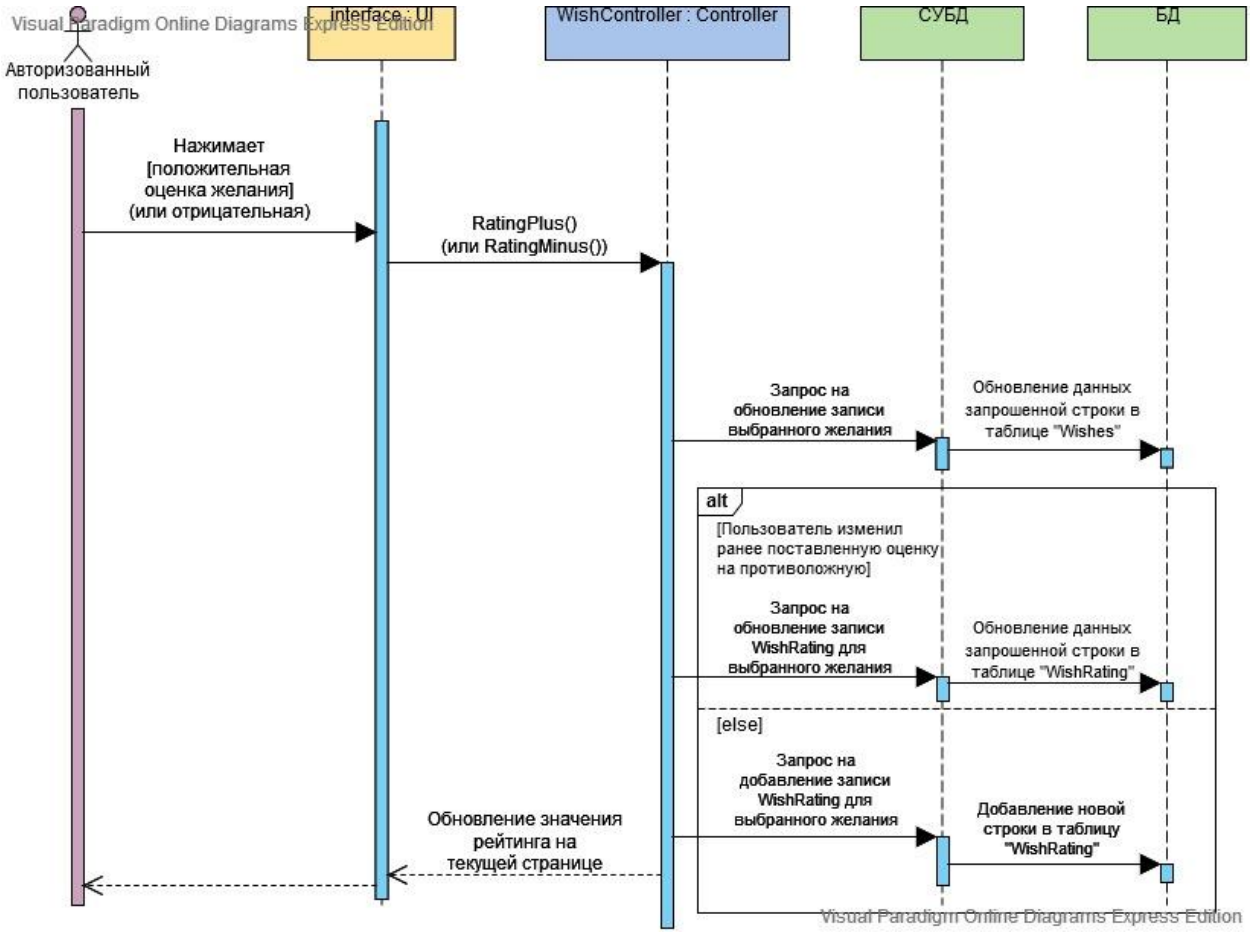

Рисунок 3. Диаграмма последовательности для оценки желания

После взаимодействии пользователя, который обязательно должен быть авторизован, с интерфейсом приложения (нажатием соответствующей кнопки для оценки желания) вызывается метод RatingPlus или RatingMinus в зависимости от выбранной оценки (положительной или отрицательной) в контроллере. Отравляется запрос в СУБД на обновление записи выбранного желания в таблице «Wishes». В результате обновляются данные строки с запрошенным WishId в таблице «Wishes» в базе данных.

Если пользователь уже ставил ранее оценку данному желанию и поставил теперь противоположную, то отравляется запрос в СУБД на обновление записи с определённым WishId в таблице «WishRating». В результате обновляются данные строки с запрошенным WishId в таблице «WishRating» в базе данных.

Если пользователь в первый раз оценил данное желание, то отравляется запрос в СУБД на добавление записи в таблице «WishRating». В результате добавляется новая строка с переданными данными в таблицу «WishRating» в базе данных.

Происходит обновление значения рейтинга на текущей странице.

Диаграмма последовательности для добавления комментария, рисунок 4

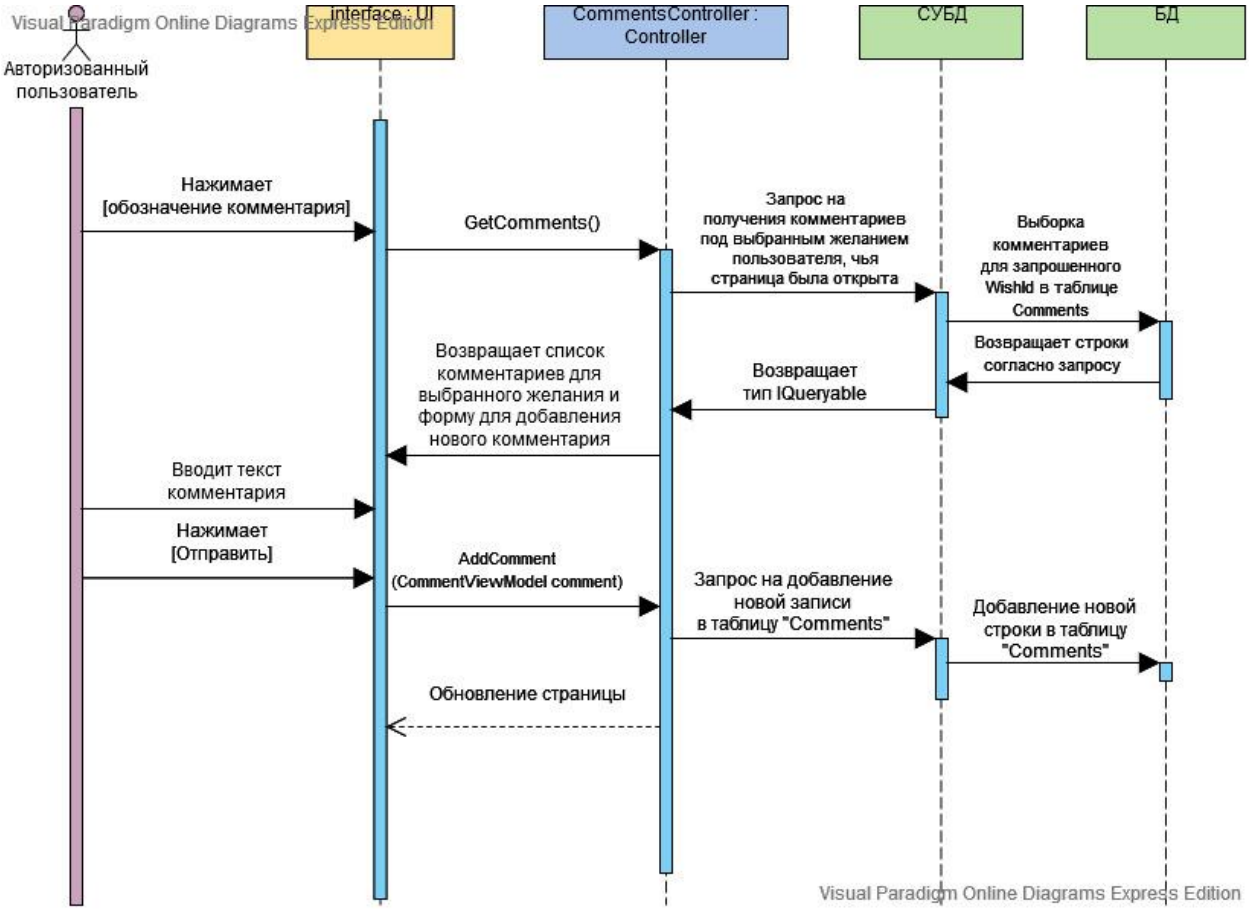

Рисунок 4. Диаграмма последовательности для добавления комментария

После взаимодействии пользователя, который обязательно должен быть авторизован, с интерфейсом приложения (нажатием кнопки, обозначающей комментарий) вызывается метод GetComments в контроллере. Отравляется запрос в СУБД на получение комментариев для выбранного желания пользователя, страница которого была открыта. В результате происходит выборка комментариев для запрошенного WishId в таблице «Comments» в базе данных. СУБД возвращается в контроллер данные типа IQueryable, т.е. множество комментариев. Полученный список отображается на странице с формой для добавления нового комментария.

После того, как пользователь ввёл содержание комментария в соответствующее поле в появившейся форме и начал кнопку «Отправить», вызывается метод AddComment в контроллере. отравляется запрос в СУБД

на добавление записи в таблице «Comments». В результате добавляется новая строка с переданными данными в таблицу «Comments» в базе данных.

Происходит обновление страницы.

Диаграмма последовательности для отметки «Подарить», рисунок 5

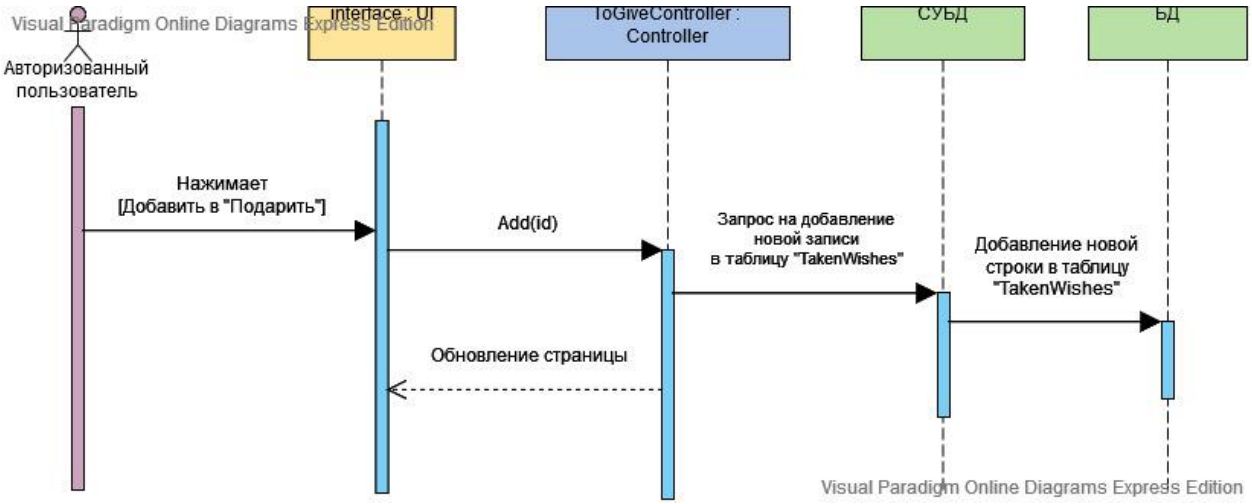

Рисунок 5. Диаграмма последовательности для отметки «Подарить»

После взаимодействии пользователя, который обязательно должен быть авторизован, с интерфейсом приложения (нажатием кнопки «добавить в “подарить”») вызывается метод Add в контроллере. Отравляется запрос в СУБД на добавление новой записи в таблицу «TakenWishes». В результате добавляется новая строка в таблицу «TakenWishes» в базе данных. Происходит обновление страницы.

Диаграмма последовательности для отметки «Подарено», рисунок 6

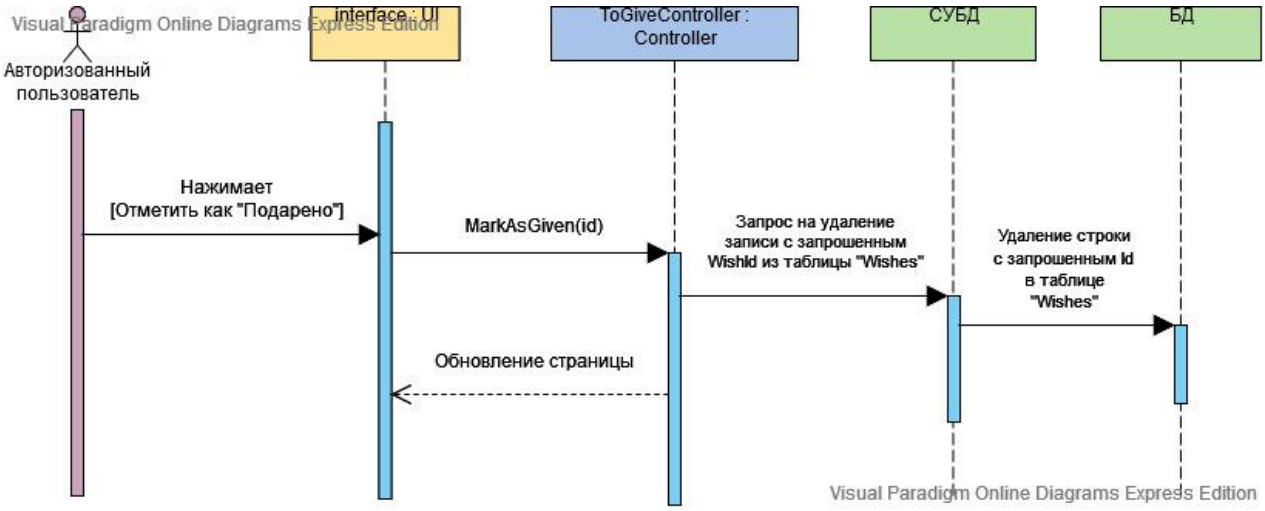

Рисунок 6. Диаграмма последовательности для отметки «Подарено»

После взаимодействии пользователя, который обязательно должен быть авторизован, с интерфейсом приложения (нажатием кнопки «отметить как “подарено”») вызывается метод MarkAsGiven в контроллере. Отравляется запрос в СУБД на удаление записи с запрошенным WishId в таблицу «Wishes». В результате удаляется строка с запрошенным WishId в таблицу «Wishes» в базе данных. Происходит обновление страницы.

2.3.3 Диаграмма взаимодействия

Диаграмма взаимодействия для создания желания, рисунок 7

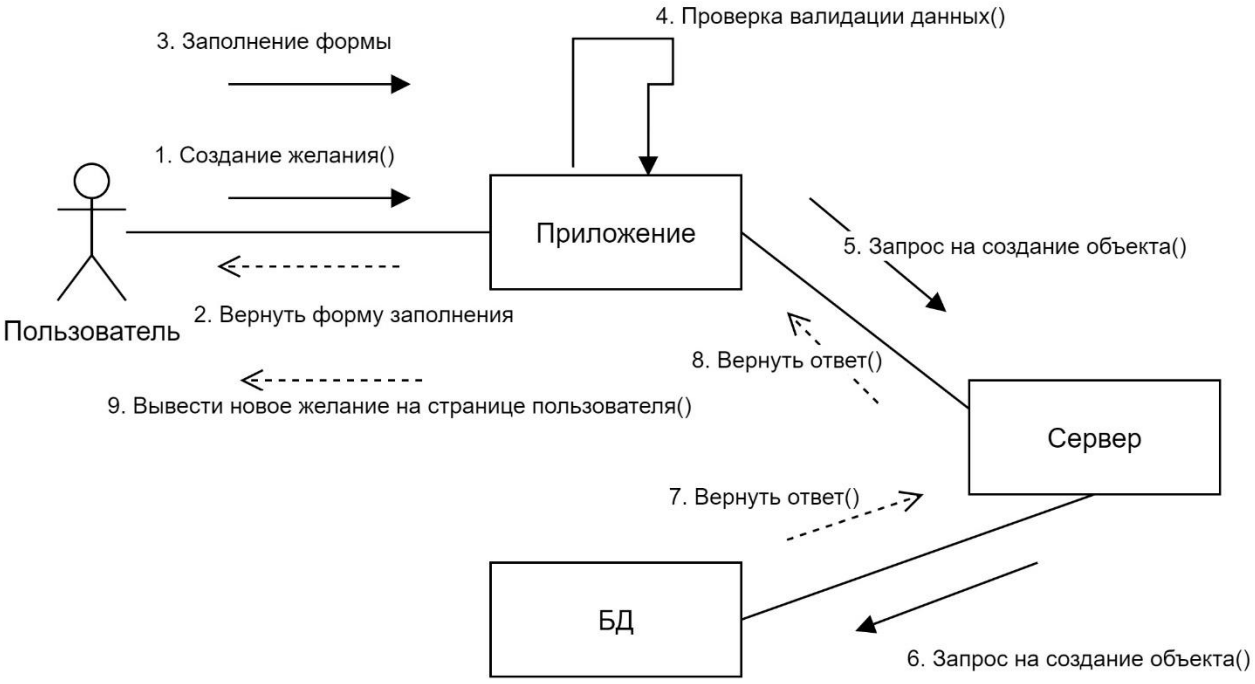

Рисунок 7. Диаграмма взаимодействия для создания желания

Для добавления нового желания пользователю необходимо на главном экране нажать кнопку «Добавить желание» (обозначена как + с подарком). После этого отображается окно для ввода данных о желании. Пользователю необходимо ввести данные и нажать кнопку «Сохранить». В случае успешно введенных данных, приложение делает запрос на сервер, который обращается в БД с запросом на сохранение. Когда данные сохранятся, сервер посылает ответ приложению. Затем пользователю отображается обновленный список желаний во вкладке “Мои желания”.

Диаграмма взаимодействия для оценки желания, рисунок 8

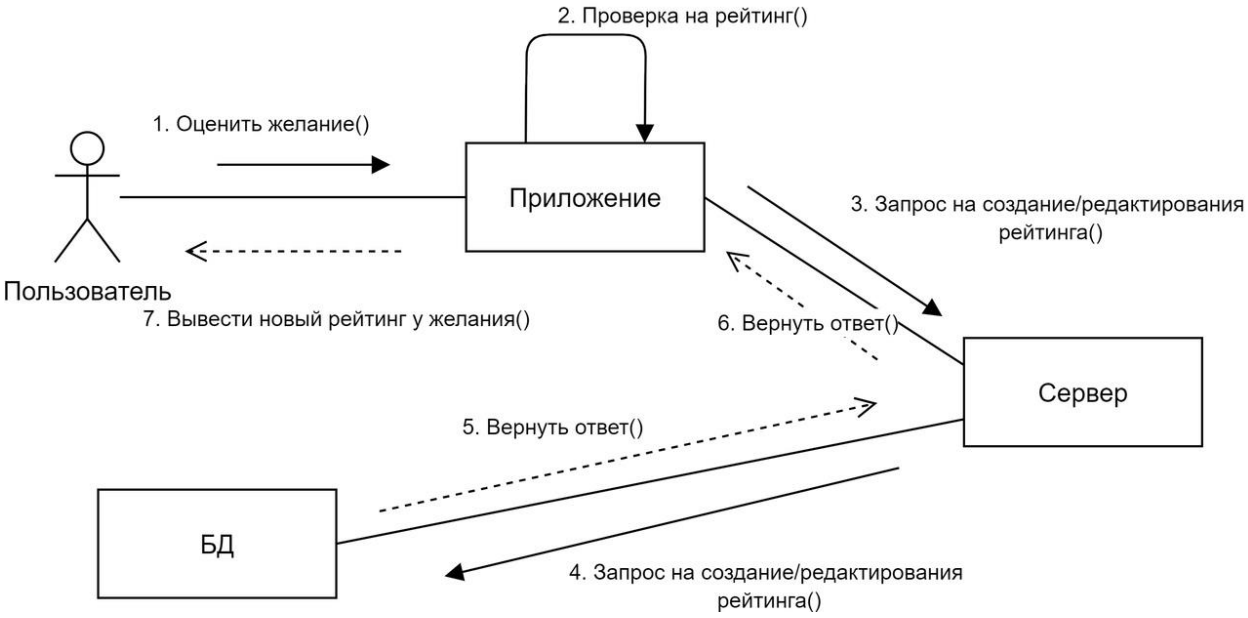

Рисунок 8. Диаграмма взаимодействия для создания желания

Для оценки желания пользователю необходимо на странице пользователя, чье желание он хочет оценить нажать на кнопки “+” или “-” соответственно тому, как он хочет оценить его. После этого приложение отправляет запрос на сервер, который обращается к БД с целью вернуть предыдущее оценивание данным пользователем этого желания. Производится проверка предыдущей оценки и в зависимости от неё приложение отправляет запрос на сервер с целью создания/редактирования предыдущей оценки. Тот в свою очередь передает запрос в БД. Если все выполнено верно, то приложение получает новый рейтинг желания, который оно и выведет.

Диаграмма взаимодействия для добавления комментария, рисунок 9

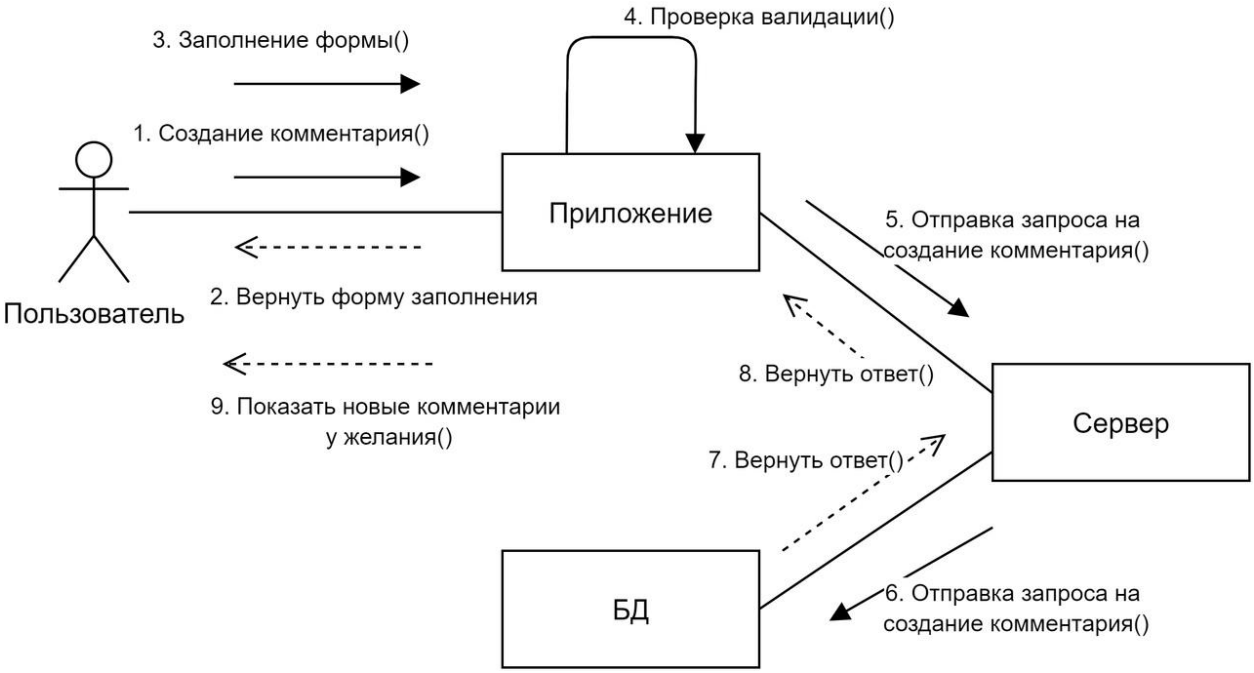

Рисунок 9. Диаграмма взаимодействия для добавления комментария

Для добавления нового комментария к желанию пользователю необходимо возле желания, под которым он хочет оставить комментарий, нажать кнопку “Комменты”. После этого отображается строка для ввода комментария. Пользователю необходимо ввести данные и нажать кнопку «Отправить». В случае успешно введенных данных, приложение делает запрос на сервер, который обращается в БД с запросом на сохранение. Когда данные сохранятся, сервер посылает ответ приложению. Затем пользователю отображается обновленный список комментариев. Если пользователю необходимо ответить на чей-то комментарий, то с начала он должен нажать кнопку “Ответить” возле того комментария, которому он хочет ответить, после чего вся вышеописанная процедура повторяется.

Диаграмма взаимодействия для отметки «Подарить», рисунок 10

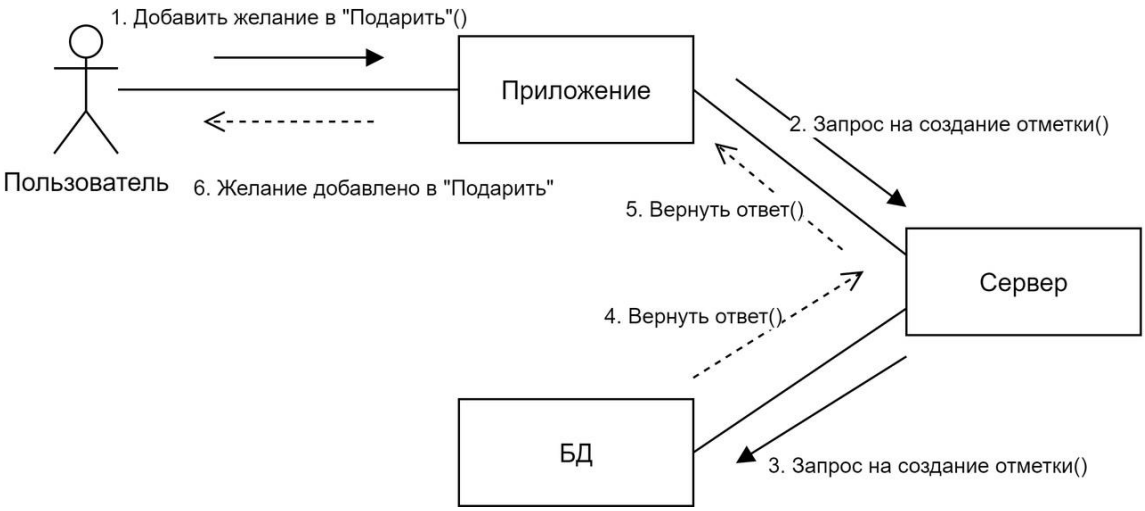

Рисунок 10. Диаграмма взаимодействия для добавления комментария

Для добавления желания в “Подарить” пользователю необходимо у желания на странице того пользователя, которому он хочет подарить нажать кнопку «Добавить в Подарить». После этого приложение делает запрос на сервер, который обращается в БД с запросом на создание отметки. Когда данные сохранятся, сервер посылает ответ приложению. Затем пользователю отображается обновленный список подарков во вкладке “Подарить”.

Диаграмма взаимодействия для отметки «Подарено», рисунок 11

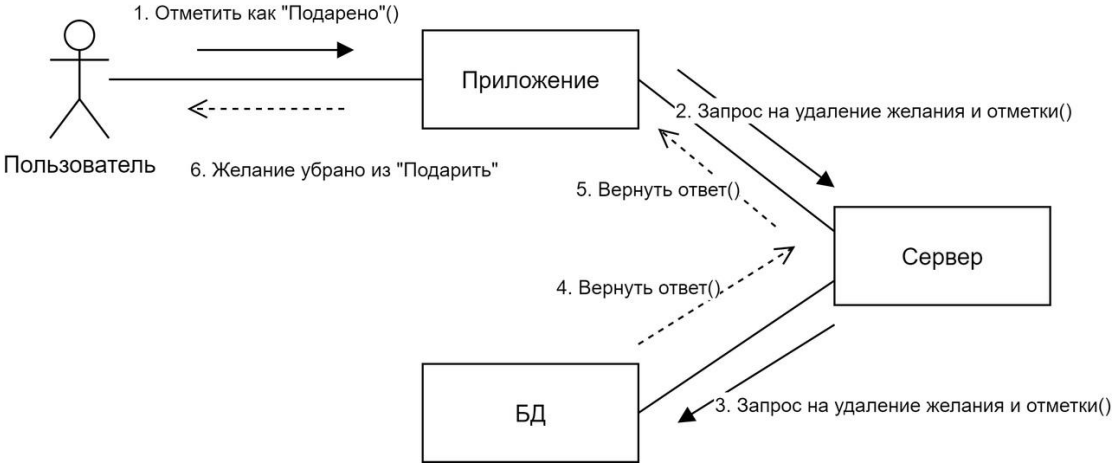

Рисунок 11. Диаграмма взаимодействия для добавления комментария

Для отметки желания как “Подарено” пользователю необходимо во вкладке “Подарить” рядом с желанием, которое он подарил нажать кнопку «Отметить как “Подарено”». После этого приложение делает запрос на сервер, который обращается в БД с запросом на удаление отметки о подарке и самого желания. Когда данные сохранятся, сервер посылает ответ приложению. Затем пользователю отображается обновленный список подарков во вкладке “Подарить”.

2.3.4 Диаграмма состояний

На рисунке 12 изображена диаграмма состояний.

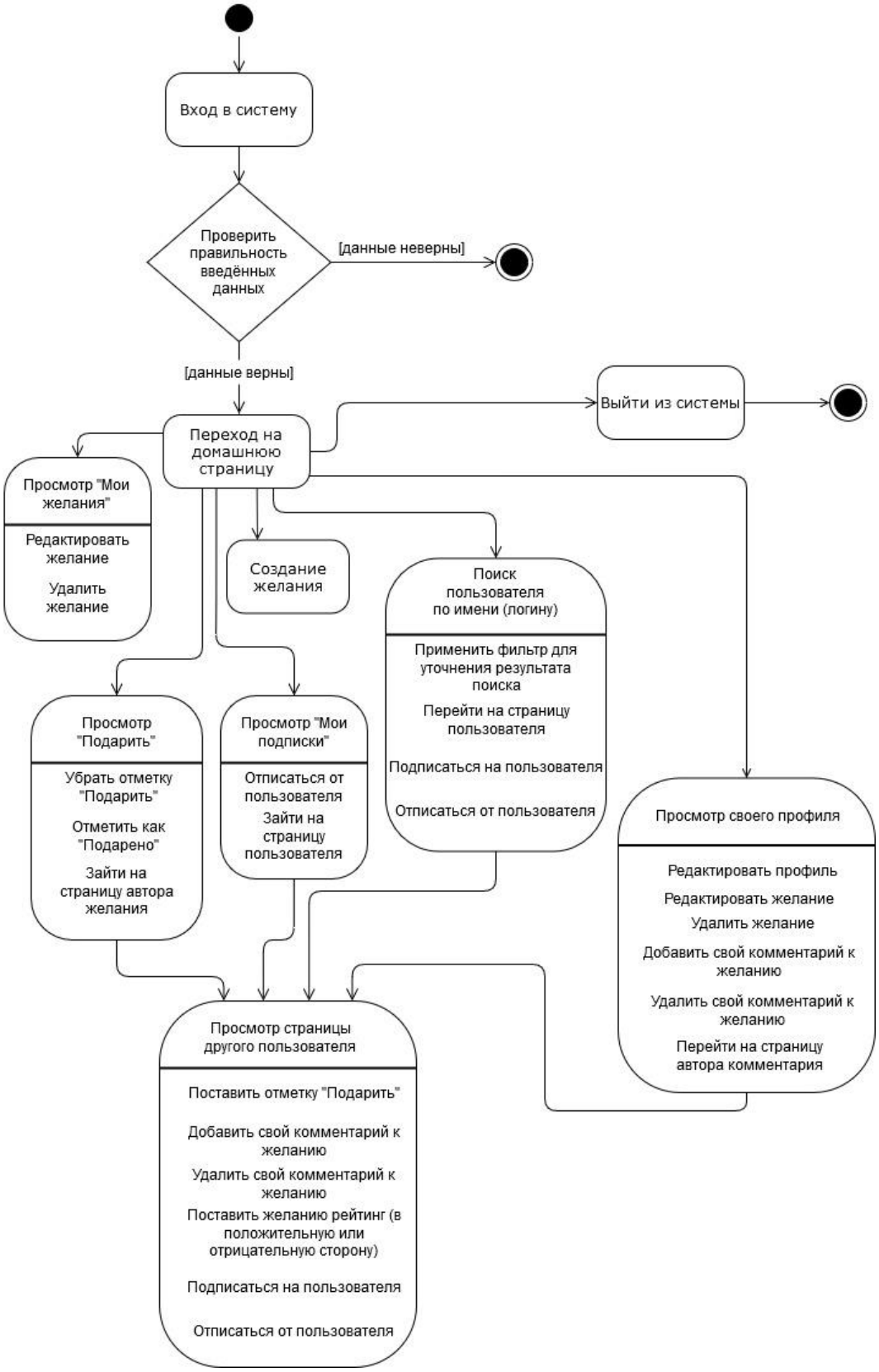

Рисунок 12. Диаграмма состояний

Диаграмма состояний отражает возможные состояния системы. При входе в систему осуществляется проверка корректности данных, которые ввёл пользователь для входа. В зависимости от результата проверки возможны 2 цепочки состояний, связанных с открытием доступа к функциональности приложения для пользователя.

Если результат проверки показал, что данные верны, система переадресовывает пользователя на домашнюю страницу, где для него доступна необходимая функциональность сайта.

Если пользователь выбрал функцию показа своего профиля, система обрабатывает запрос, отображает пользователю запрошенную страницу, где текущему пользователю предоставляется соответствующий данному состоянию перечень функций.

Аналогично происходит и с другими цепочками состояний.

Из любого состояния системы, кроме входа в систему, пользователь может выйти из системы.

2.3.5 Диаграмма развёртывания

На рисунке 13 представлена диаграмма развёртывания, чтобы определить, какие аппаратные компоненты («узлы») существуют, какие программные компоненты работают на каждом узле и какие различные части этого комплекса соединяются друг с другом.

Для развёртывания приложения требуется сервер с высокоскоростным доступом в интернет. Пользователю для взаимодействия с приложением требуется персональный компьютер с доступом в интернет, а также устройствами ввода-вывода информации, такие как клавиатура и мышь или сенсорная панель и монитор (экран).

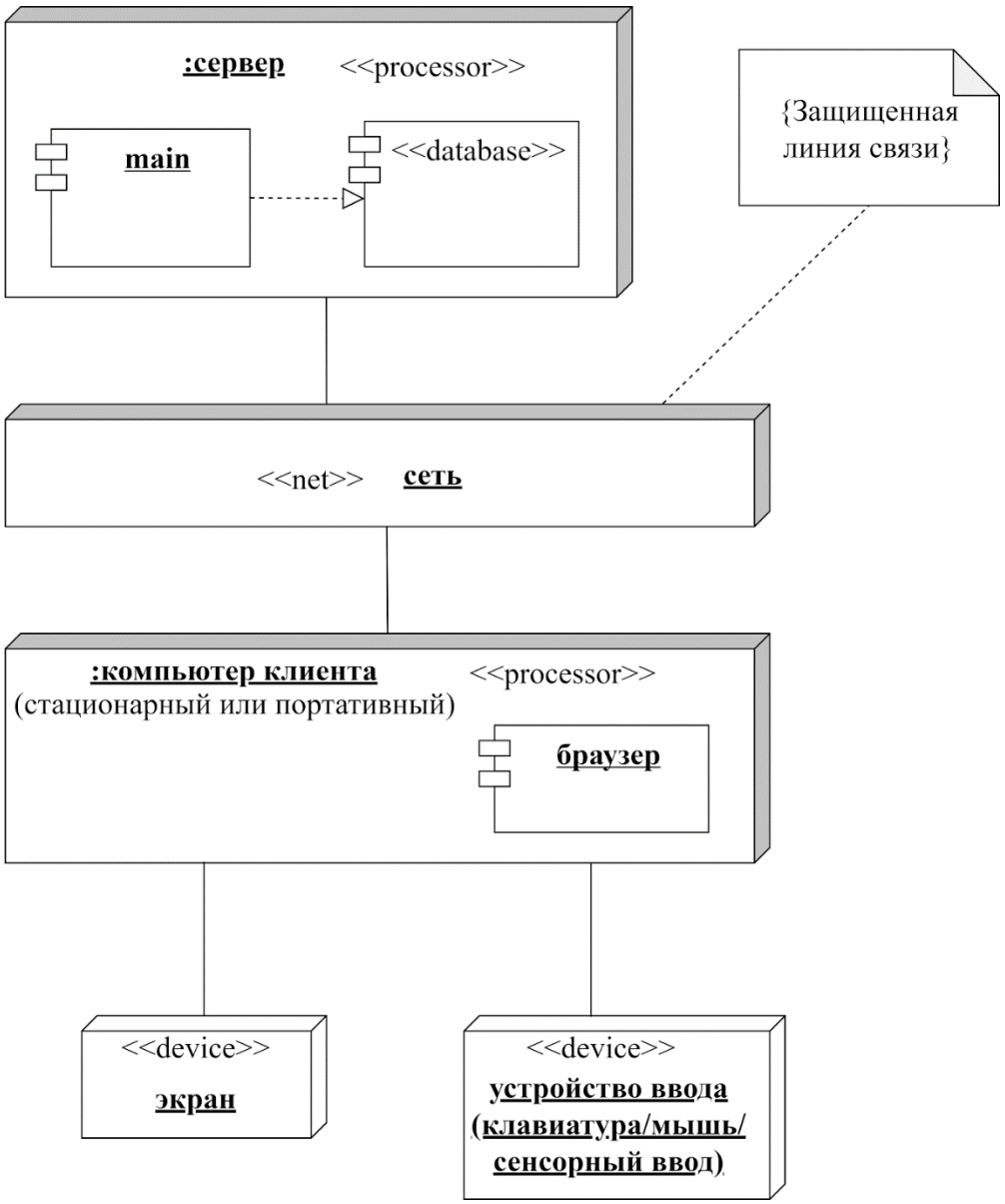

Рисунок 13. Диаграмма развёртывания

2.3.6 Диаграмма объектов

На рисунке 14 представлена диаграмма объектов

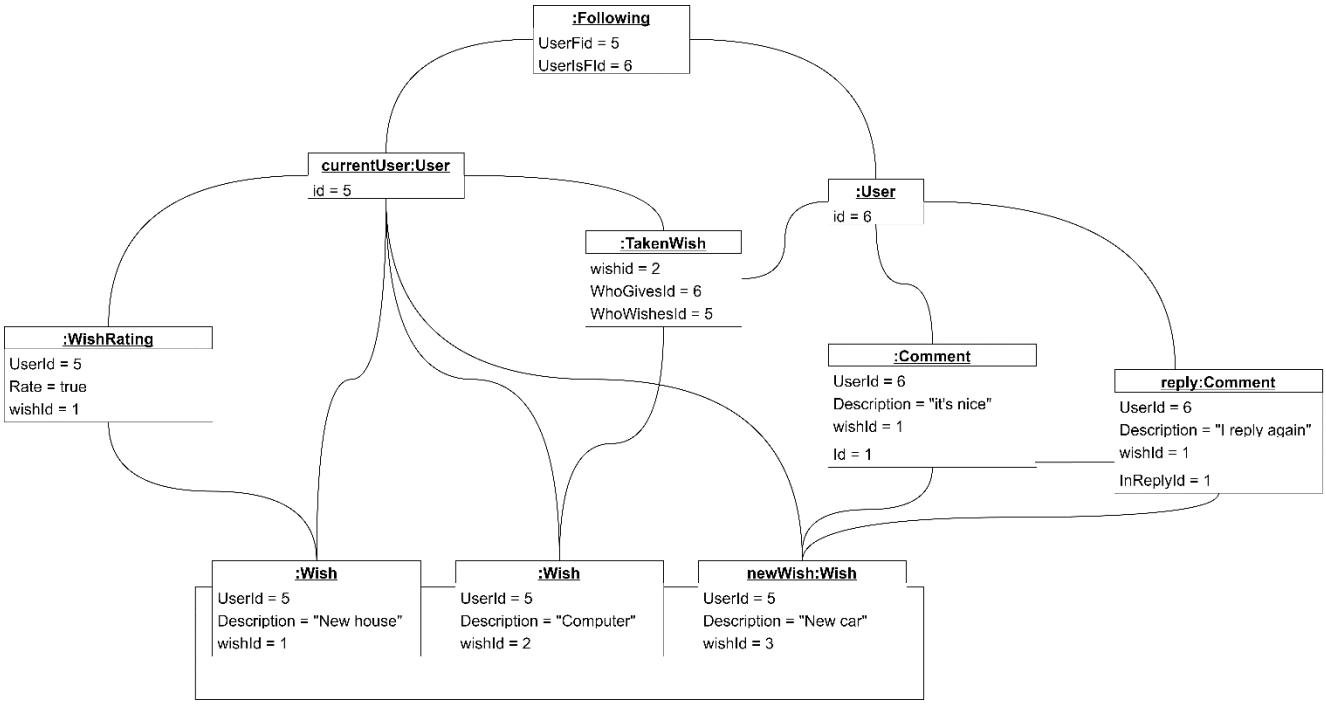

Рисунок 14. Диаграмма объектов

Пусть в некоторый момент времени существуют следующие объекты:

- экземпляр класса User с именем currentUser выступающий в роли текущего авторизованного пользователя
- 3 экземпляра “желаний” currentUser’а
- анонимный экземпляр класса User
- экземпляр класса WishRating, хранящий информацию о том, что текущий пользователь нажал на кнопку “+” рейтинга своего желания
- экземпляр класса Following, хранящий информацию о том, что пользователь с Id = 6 подписался на новости currentUser
- экземпляр класса TakenWish, хранящий информацию о том, что пользователь с Id = 6 решил подарить одно из желаний currentUser и нажал соответствующую кнопку
- анонимный экземпляр класса Comment, хранящий информацию о комментарии, оставленным пользователем с Id = 6 под желанием пользователя currentUser

2.3.7 Диаграмма активности

На рисунке 15 изображена диаграмма активности

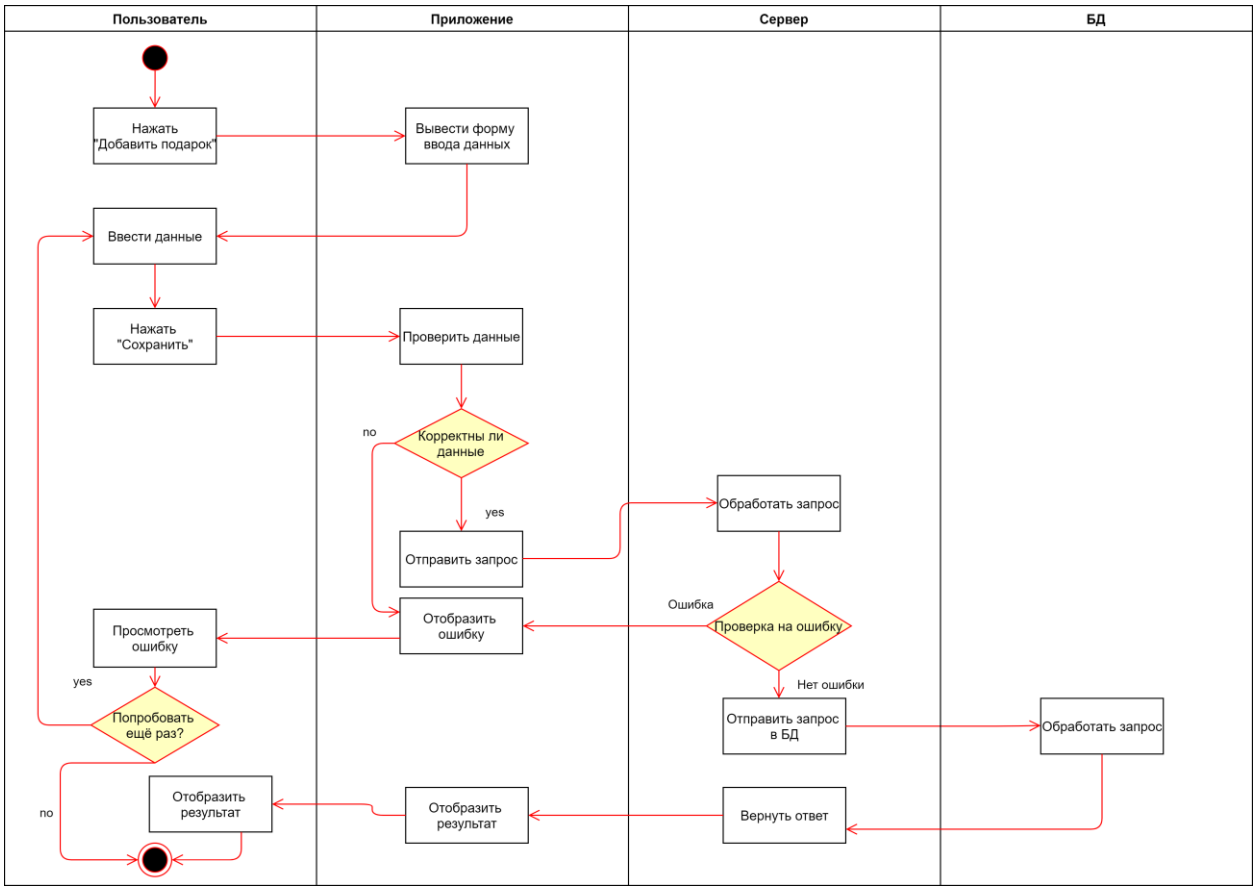

Рисунок 15. Диаграмма активности для создания желания

Авторизованный пользователь может добавить новое желание с главного экрана, нажав кнопку «Добавить желание» (обозначена как + с подарком). Приложение отобразит окно для ввода данных. Пользователю необходимо ввести данные о новом желании и нажать кнопку «Сохранить».

Далее приложение проверяет введенные данные, если данные корректны, приложение отправляет запрос на сервер. Если пользователь ввел неверные данные, приложение выведет пользователю ошибку. После чего пользователь может заново ввести данные или закончить действие. После отправки запроса на сервер, сервер обрабатывает запрос и отправлять запрос к базе данных. После чего получает данные от БД и возвращает ответ. Приложение возвращает результат пользователю, и пользователь просматривает результат.

Если при обработке на сервере произошла ошибка или ответ от сервера не получен, приложение отображает пользователю ошибку, пользователь просматривает ошибку, после чего может попробовать еще раз или завершить действие.

2.3.8 Диаграмма классов

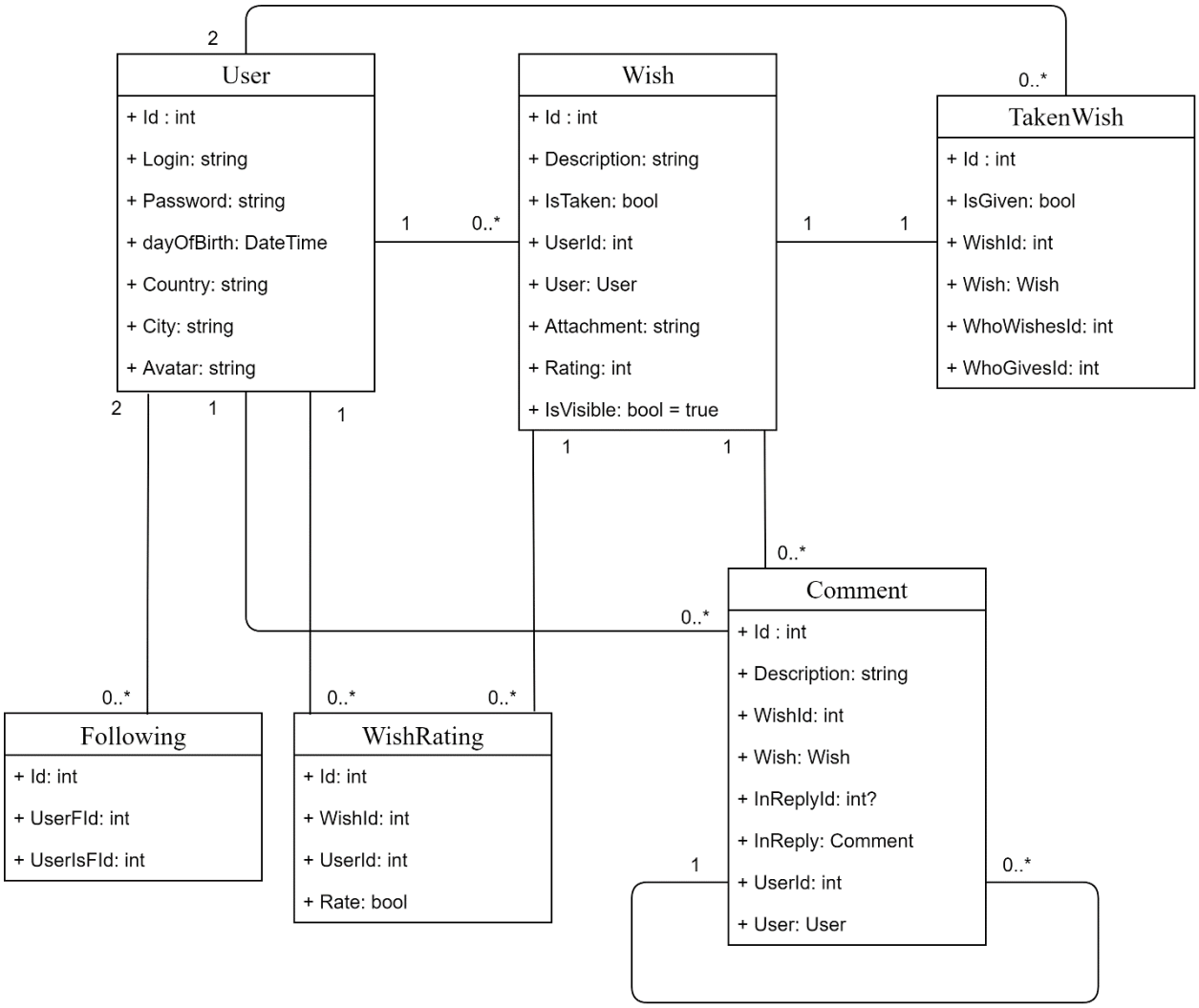

Рисунок 16. Диаграмма классов

Имеются несколько классов:

User, обозначающий пользователя приложения:

id - уникальный идентификатор

login - имя пользователя в системе

password - пароль

dayOfBirth - дата рождения

Country - страна проживания

City - город

Avatar - аватар пользователя;

Wish, обозначающий желания пользователей:

id - уникальный идентификатор

description - описание желания

IsTaken - хочет ли какой-то другой пользователь подарить этот предмет

UserId - уникальный идентификатор автора данного желания

User - автор желания

Attachment - изображение желания

Rating - рейтинг данного желания

IsVisible - видно ли это желание в приложении. Если у желания низкий рейтинг, то оно скрывается. По умолчанию все желания видны;

Following, обозначающий подписки пользователя на других пользователей:

id - уникальный идентификатор

UserFId - пользователь, который подписывается

UserIsFId - пользователь, на которого подписываются;

WishRating, обозначающий рейтинг(+ или -) того или иного желания от конкретного пользователя:

id - уникальный идентификатор

WishId - уникальный идентификатор желания

UserId - пользователь, который оценивает желание

Rate - сама оценка(“+” это true, “-” это false). Сделано для того, чтобы один пользователь не мог несколько раз поставить отрицательную или положительную оценку желанию;

WishRating был введён с целью контроля контента, который публикуют пользователь. Если рейтинг желания достигнет определённого порога, то оно будет удалено в связи с тем, что по мнению других пользователей представляет неуместный контент.

Comment, обозначающий комментарий под тем или иным желанием:

Id - уникальный идентификатор

Description - сам комментарий

WishId - уникальный идентификатор желания, под которым оставляется комментарий

Wish - само желание

InReply - дописать

InReplyId - дописать

UserId - уникальный идентификатор автора комментария

User - автор комментария;

TakenWish создан для сохранения информации о том, какой пользователь поставил отметку «Подарить» определённому желанию другого пользователя:

Id - уникальный идентификатор

IsGiven – флаг о том, что подарок был вручён автору желания

WishId - уникальный идентификатор желания, которое было отмечено как «подарить»

Wish - само желание

WhoWishesId - уникальный идентификатор автора желания

WhoGivesId – уникальный идентификатор пользователя, который хочет подарить данный подарок.

2.4 Воронки в Яндекс Метрике.

2.4.1 Воронка “Создание желания”

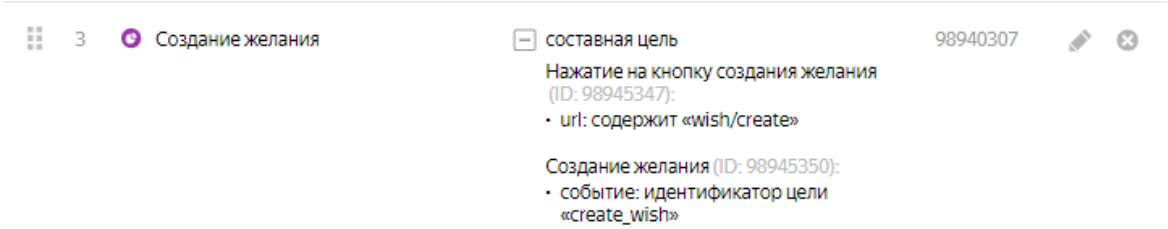

Рисунок 17. Настройка воронки «Создание желания»

Рисунок 18. Графическая иллюстрация воронки «Создание желания»

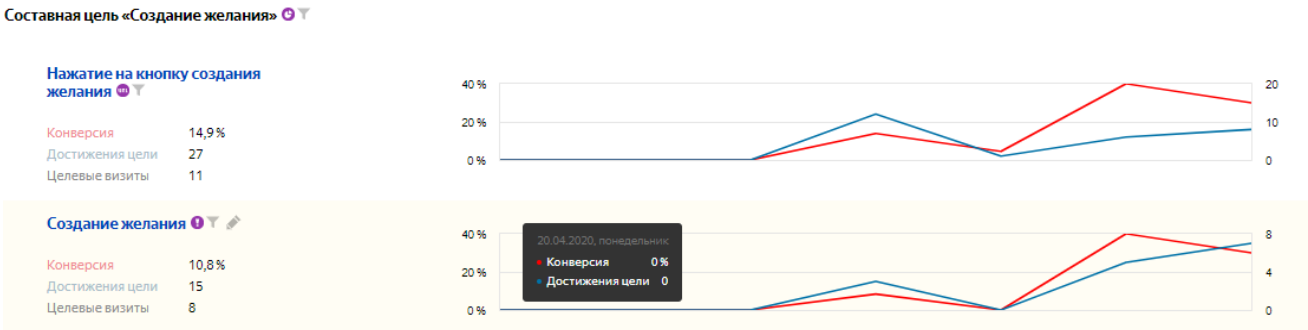

Рисунок 19. Конверсия воронки «Создание желания»

Для выполнения условий данной воронки пользователь должен последовательно выполнить 2 действия:

- перейти на страничку в URL адресе которой содержится строка “/wish/create”
- нажать на кнопку “Сохранить”, которая вызывает выполнение JS скрипта, отправляющего данные о достижении заданной цели в Яндекс Метрику.

2.4.2 Воронка «Оценка желаний»

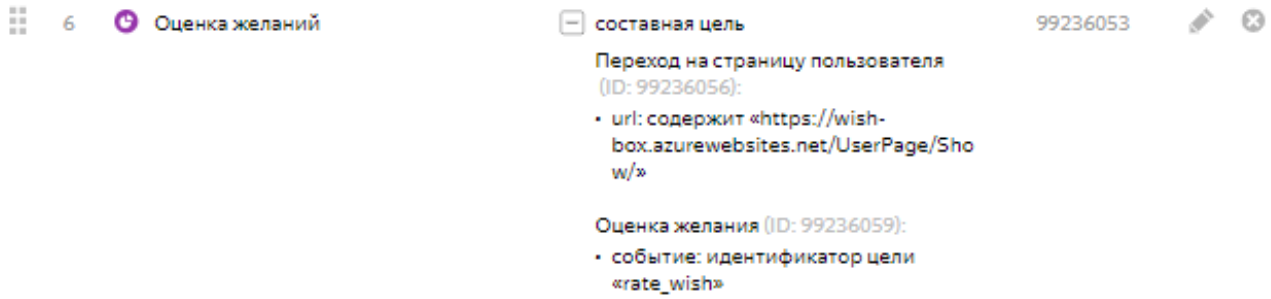

Рисунок 20. Настройка воронки «Оценка желаний»

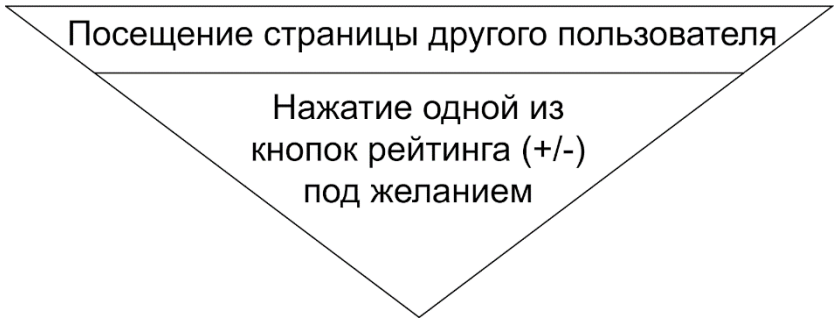

Рисунок 21. Графическая иллюстрация воронки «Оценка желаний»

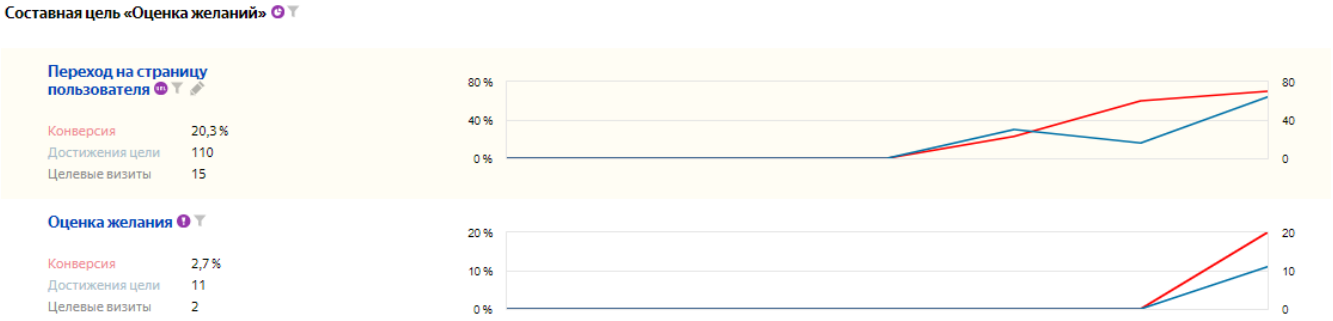

Рисунок 22. Конверсия воронки «Оценка желаний»

Для выполнения условий данной воронки пользователь должен последовательно выполнить 2 действия:

- перейти на страничку в URL адресе которой содержится строка “https://wish-box.azurewebsites.net/UserPage/Show/”
- нажать на одну из кнопок (+ или -) расположенных рядом с каждым желанием (если текущая страница не является собственной страницей пользователя), которая вызывает выполнение JS скрипта, отправляющего данные о достижении заданной цели в Яндекс Метрику.

2.4.3 Воронка «Добавить комментарий»

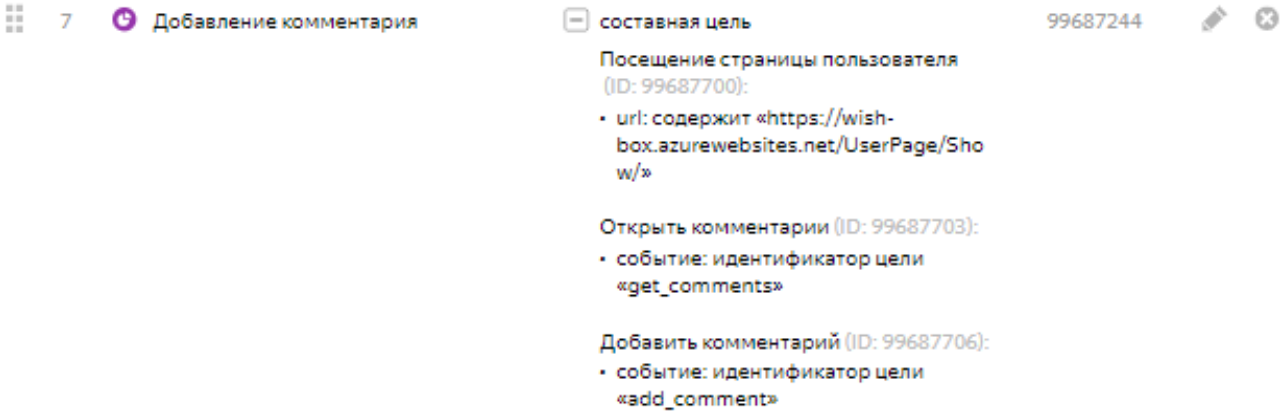

Рисунок 23. Настройка воронки «Добавить комментарий»

Рисунок 24. Графическая иллюстрация воронки «Добавить комментарий»

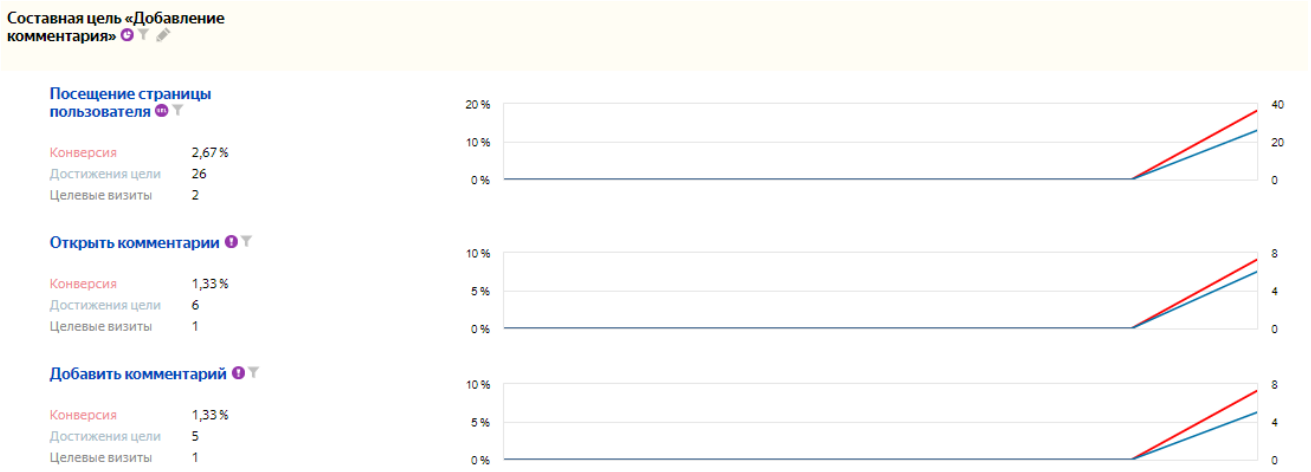

Рисунок 25. Конверсия воронки «Добавить комментарий»

Для выполнения условий данной воронки пользователь должен последовательно выполнить 3 действия:

- перейти на страничку в URL адресе которой содержится строка “https://wish-box.azurewebsites.net/UserPage/Show/”
- нажать на одну из кнопок “Комменты” расположенных рядом с каждым желанием, которая вызывает выполнение JS скрипта, отправляющего данные о достижении заданной цели в Яндекс Метрику.
- нажать на кнопку “Отправить”, которая вызывает выполнение JS скрипта, отправляющего данные о достижении заданной цели в Яндекс Метрику.

2.4.4 Воронка «Отметка “Подарить”»

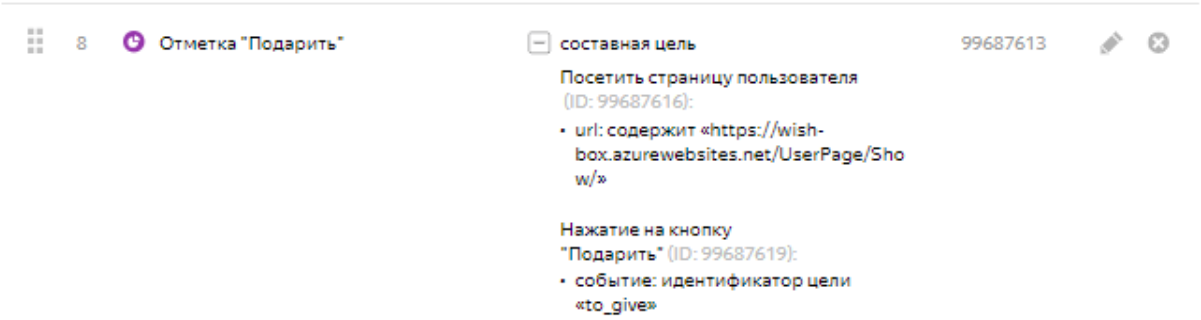

Рисунок 26. Настройка воронки «Отметка “Подарить”»

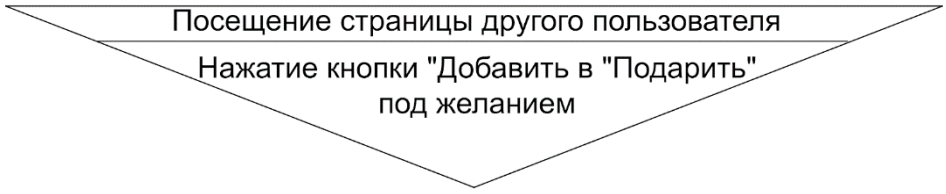

Рисунок 27. Графическая иллюстрация воронки «Отметка “Подарить”»

Рисунок 28. Конверсия воронки «Отметка “Подарить”»

Для выполнения условий данной воронки пользователь должен последовательно выполнить 2 действия:

- перейти на страничку в URL адресе которой содержится строка “https://wish-box.azurewebsites.net/UserPage/Show/”
- нажать на одну из кнопок “Добавить в “Подарить” расположенных рядом с каждым желанием (если текущая страница не является собственной страницей пользователя), которая вызывает выполнение JS скрипта, отправляющего данные о достижении заданной цели в Яндекс Метрику.

2.4.5 Воронка «Отметить как “Подарено”»

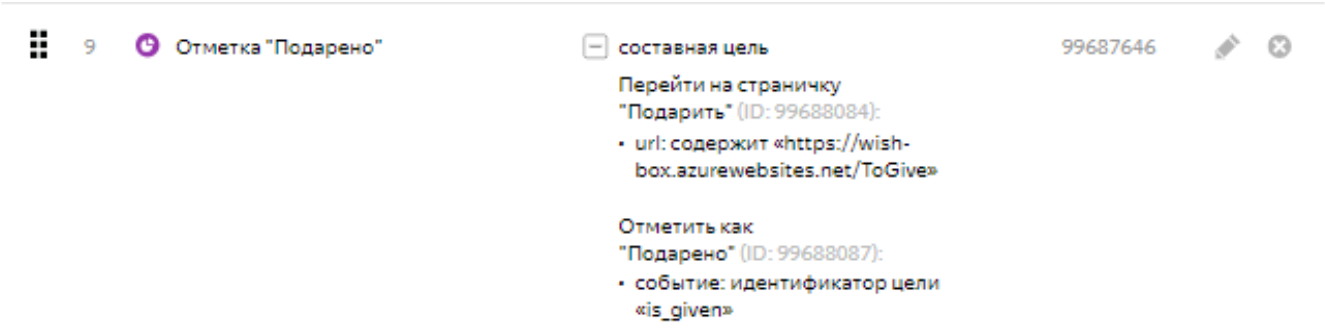

Рисунок 29. Настройка воронки «Отметить как “Подарено”»

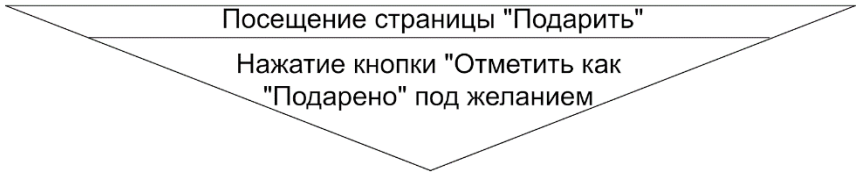

Рисунок 30. Графическая иллюстрация воронки «Отметить как “Подарено”»

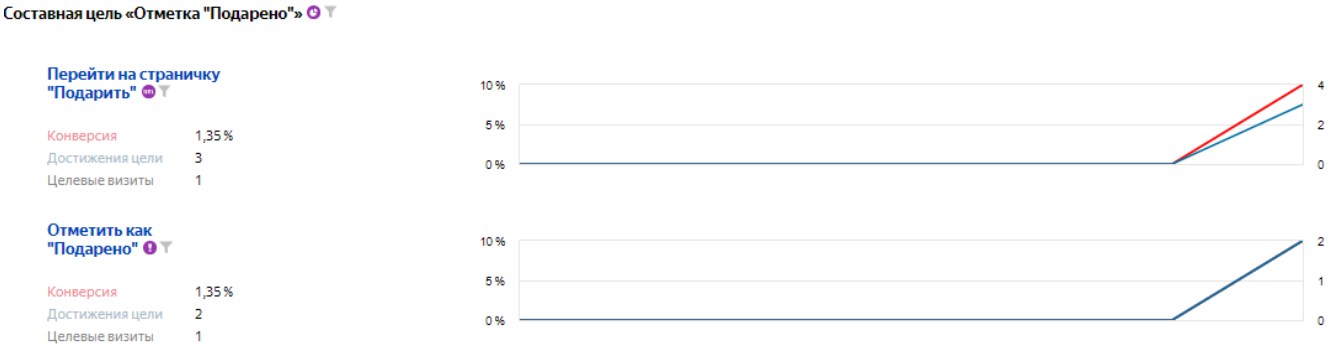

Рисунок 31. Конверсия воронки «Отметить как “Подарено”»

Для выполнения условий данной воронки пользователь должен последовательно выполнить 2 действия:

- перейти на страничку в URL адресе которой содержится строка “https://wish-box.azurewebsites.net/ToGive”
- нажать на одну из кнопок “Отметить как “Подарено” расположенных рядом с каждым желанием, которая вызывает выполнение JS скрипта, отправляющего данные о достижении заданной цели в Яндекс Метрику.

2.5 Схема базы данных

На рисунке 32 представлена ER-диаграмма, на которой показаны основные сущности системы и их взаимосвязь между собой.

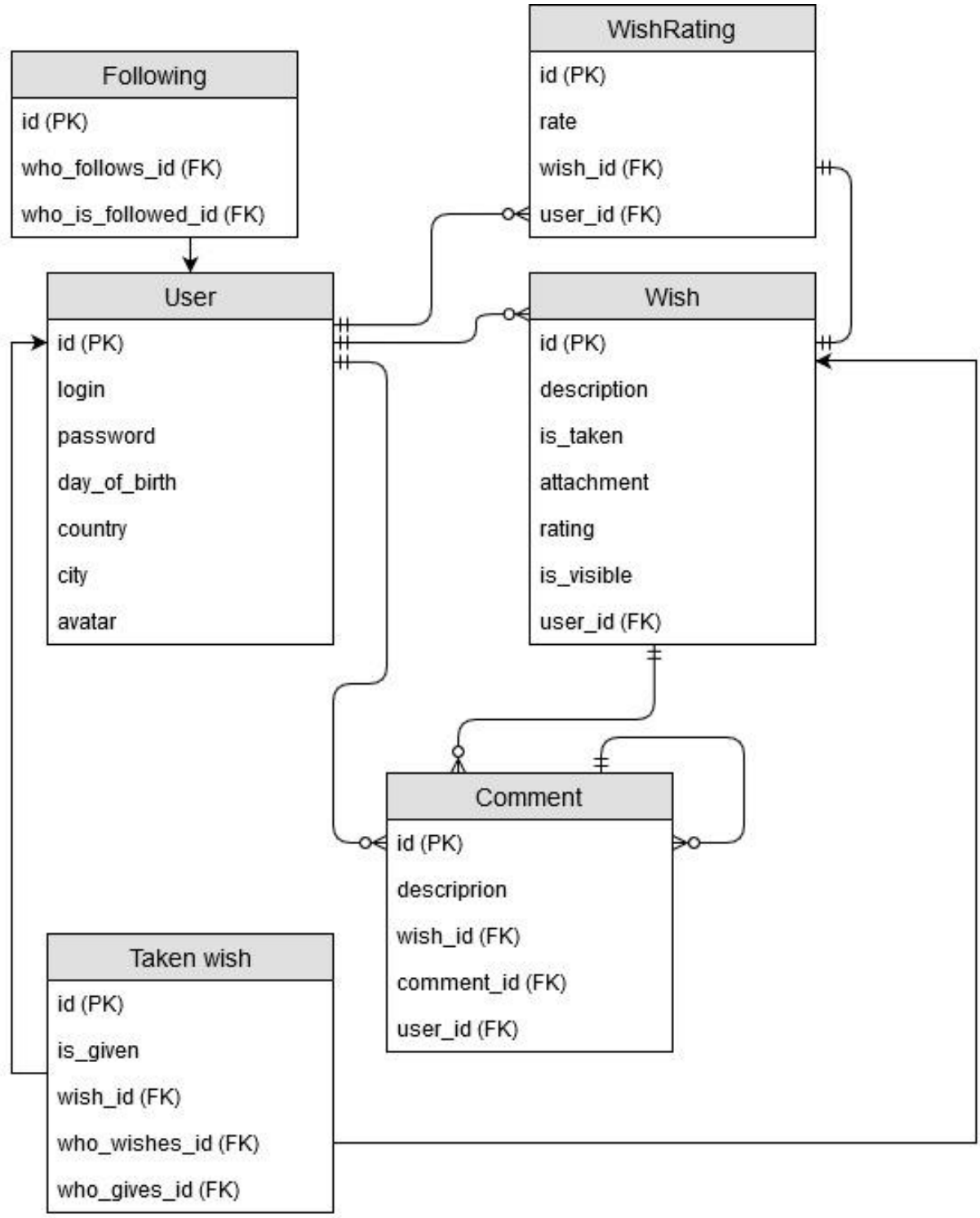

Рисунок 32. ER-диаграмма

- У пользователя может быть одно или несколько желаний (или не быть их совсем), а желания обязательно должен быть только один пользователь. Т.к. связь 1:m, в сущности “Wish” содержится внешний ключ “user_id”, который будет указывать на id пользователя, который создал данное желание.
- Т.к. к желанию можно написать комментарий, получается, что у желания может быть один или несколько комментариев (или может не быть вовсе), а комментарий должен обязательно ссылаться только на одно желание и только на одного пользователя. В функциональности так же предусмотрен ответ на другой комментарий, поэтому будет существовать связь между сущностями “Comment” как 1:m, где у комментария может быть один или несколько ответных комментарий (или может не быть вовсе), а ответный комментарий обязательно должен ссылаться только на один комментарий.
- “Taken wish” – это соединяющая таблица для “User” и “Wish”, где хранятся данные о том, какое желание было отмечено, каким пользователем, кто создал данное желание и было ли оно подарено.
- “Following” – таблица, хранящая данные о том, какой пользователь на кого подписан.
- “WishRating” – таблица, хранящая данные о том, какой рейтинг (в большую сторону или в меньшую) поставил определённый пользователь определённому желанию другого пользователя. Конкретный рейтинг может относится только к одному желания, а пользователь может поставить рейтинг для многих желаний, а оценка пользователя относится только к этому конкретному пользователю.

3. Обоснование архитектуры проекта

3.1 Определение архитектуры проекта

В качестве архитектуры приложения был взят паттерн MVC - Model-View-Controller (рисунок 33).

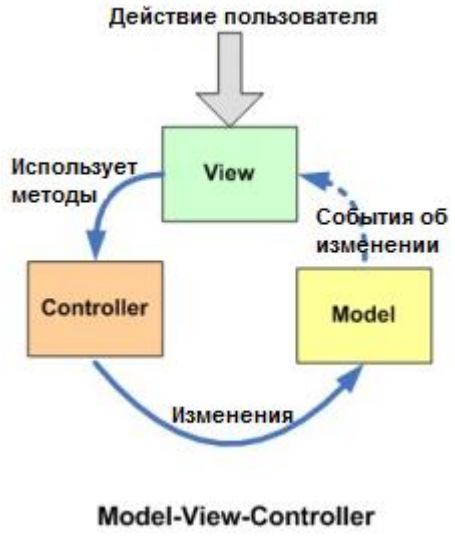

Рисунок 33. Схема работы MVC

Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.

Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.

Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Основная цель применения этой концепции состоит в отделении бизнес-логики (модели) от её визуализации (представления, вида). За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в

различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

1) К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы;
2) Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных) — для этого достаточно использовать другой контроллер;
3) Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

3.2 Сравнение с другими паттернами

Рассмотрим различия между паттерном MVC и паттернами MVP, MVVM.

MVP - Model-View-Presenter (рисунок 34).

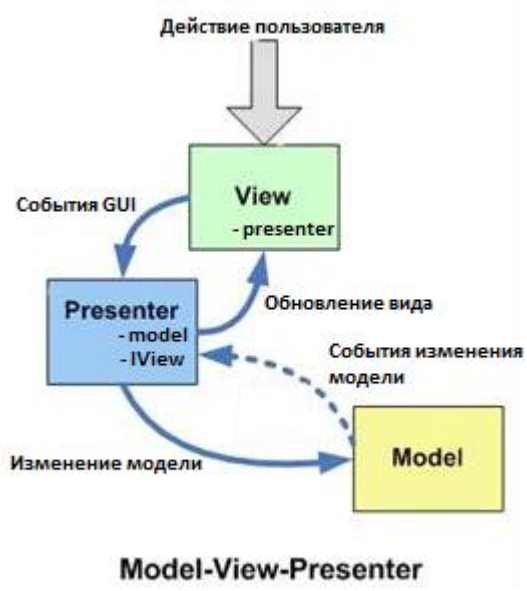

Рисунок 34. Схема работы MVP

Данный подход позволяет создавать абстракцию представления. Для этого необходимо выделить интерфейс представления с определенным набором свойств и методов. Презентер, в свою очередь, получает ссылку на реализацию интерфейса, подписывается на события представления и по запросу изменяет модель.

Признаки презентера:

- Двухсторонняя коммуникация с представлением;
- Представление взаимодействует напрямую с презентером, путем вызова соответствующих функций или событий экземпляра презентера;
- Презентер взаимодействует с View путем использования специального интерфейса, реализованного представлением;
- Один экземпляр презентера связан с одним отображением.

Реализация:

Каждое представление должно реализовывать соответствующий интерфейс. Интерфейс представления определяет набор функций и событий, необходимых для взаимодействия с пользователем (например, IView.ShowErrorMessage(string msg)). Презентер должен иметь ссылку на реализацию соответствующего интерфейса, которую обычно передают в конструкторе.

Логика представления должна иметь ссылку на экземпляр презентера. Все события представления передаются для обработки в презентер и практически никогда не обрабатываются логикой представления (в т.ч. создания других представлений).

Пример использования: Windows Forms.

Правила применения:

- Используется в ситуации, когда невозможно связывание данных (нельзя использовать Binding);
- Частым примером может быть использование Windows Forms.

MVVM - Model-View-View Model (рисунок 35).

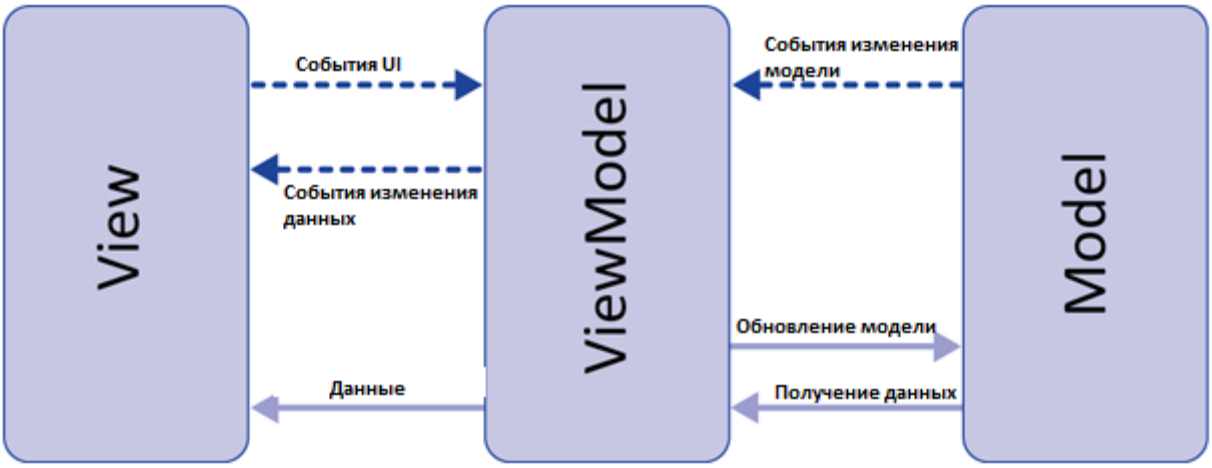

Рисунок 35. Схема работы MVVM

Данный подход позволяет связывать элементы представления со свойствами и событиями View-модели. Можно утверждать, что каждый слой этого паттерна не знает о существовании другого слоя.

Признаки View-модели:

- Двухсторонняя коммуникация с представлением;
- View-модель — это абстракция представления. Обычно означает, что свойства представления совпадают со свойствами View-модели / модели
- View-модель не имеет ссылки на интерфейс представления (IView). Изменение состояния View-модели автоматически изменяет представление и наоборот, поскольку используется механизм связывания данных (Bindings)
- Один экземпляр View-модели связан с одним отображением.

Реализация:

При использовании этого паттерна, представление не реализует соответствующий интерфейс (IView).

Представление должно иметь ссылку на источник данных (DataContex), которым в данном случае является View-модель. Элементы представления связаны (Bind) с соответствующими свойствами и событиями View-модели.

В свою очередь, View-модель реализует специальный интерфейс, который используется для автоматического обновления элементов представления. Примером такого интерфейса в WPF может быть INotifyPropertyChanged.

Пример использования: WPF

Правила применения:

- Используется в ситуации, когда возможно связывание данных без необходимости ввода специальных интерфейсов представления (т.е. отсутствует необходимость реализовывать IView);
- Частым примером является технология WPF.

3. Инструменты разработки

4. Описание работы программы

4.1 **Описание клиентской части приложения**

4.2 **Описание серверной части приложения**

5. Список использованных источников

6. Приложения